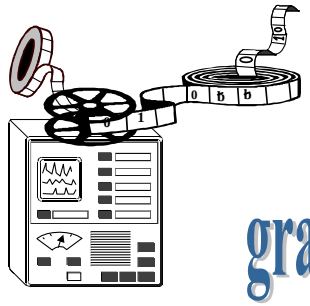
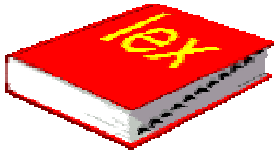
Lenguajes, gramáticas y autómatas









Alfabetos y cadenas (1)

- **Alfabeto**: Un alfabeto Σ es un conjunto *finito* y *no vacío* de símbolos.
- **Cadena sobre un alfabeto Σ**: Es una sucesión de caracteres tomados de Σ.
- **Cadena vacía**: Cadena sin elementos, se representa por ε
- **Longitud de una cadena *c***: Es el número de caracteres que forman *c*, se denota por $|c|$ o por $long(c)$

  $|\varepsilon| = long(\varepsilon) = 0$
- **Concatenación de cadenas**: La concatenación de la cadena *x* con la cadena *y* es la cadena que resulta al añadir a la cadena *x* la cadena *y*, se denota por *xy*.

Alfabetos y cadenas (2)

- Propiedades de la concatenación:
  1. Es una operación cerrada
  2. Es asociativa
  3. La cadena vacía es el elemento identidad
  4. La concatenación no es conmutativa
  5. Verifica las leyes de cancelación izquierda y derecha
- Para representar la concatenación repetida se usa una notación de tipo exponencial

$$x^n = \underbrace{xx\cdots x}_{n \text{ veces}} \qquad x^{n+1} = xx^n = x^n x \qquad x^n x^m = x^{n+m} \qquad x^1 = x \qquad x^0 = \varepsilon$$

Lenguajes

- **Lenguaje universal sobre Σ**: Conjunto de todas las posibles cadenas formadas sobre Σ (incluida ε), se denota por Σ*.
- **Lenguaje formal sobre un alfabeto Σ**: Conjunto de cadenas sobre Σ, cualquier subconjunto de Σ*.
- **Lenguaje vacío**: Es aquel lenguaje que no contiene ninguna cadena, ni siquiera la vacía. Se representa por Φ.
- Ejemplos: Dado el alfabeto $\Sigma=\{a, b\}$
  - lenguaje de las cadenas con exactamente una *b*, $L_1=\{b, ab, a^3ba^2, ...\}$
  - lenguaje de las cadenas de longitud menor o igual que uno, $L_2=\{\varepsilon, a, b\}$
  - lenguaje vacío, $L_3=\Phi$
  - lenguaje que solo tiene la cadena vacía, $L_4=\{\varepsilon\}$

Operaciones sobre lenguajes (1)

- **Unión de lenguajes**: La unión $L_1 \cup L_2$ de dos lenguajes L_1 y L_2 es el lenguaje formado por las cadenas que pertenecen bien a L_1 bien a L_2. La unión de lenguajes constituye un *monoide*.

$$L_1 \cup L_2 = \{x: x \in L_1 \vee x \in L_2\}$$

- **Intersección de lenguajes**: La intersección $L_1 \cap L_2$ de dos lenguajes L_1 y L_2 es el lenguaje formado por las cadenas que pertenecen simultáneamente a L_1 y a L_2.

$$L_1 \cap L_2 = \{ x: x \in L_1 \wedge x \in L_2\}$$

- **Diferencia de lenguajes**: La diferencia L_1-L_2 de dos lenguajes L_1 y L_2 es el lenguaje formado por las cadenas que pertenecen a L_1 y no pertenece a L_2.

$$L_1 - L_2 = \{ x: x \in L_1 \wedge x \notin L_2\}$$

Operaciones sobre lenguajes (2)

- **Complementación de un lenguaje**: La complementación $\bar{L}$ de un lenguaje L sobre un alfabeto Σ es el lenguaje formado por las cadenas que pertenecen al lenguaje universal sobre Σ (Σ*) pero que no pertenecen a L. O dicho de otro modo, esta formado por las cadenas que pertenecen a Σ*-L.

$$\bar{L} = \{x: x \in \Sigma^* \wedge x \notin L\} = \{x: x \in \Sigma^* - L\}$$

- **Concatenación**: La concatenación $L_1 \cdot L_2$ de dos lenguajes L_1 y L_2 es el lenguaje formado por cadenas construidas mediante la concatenación de una cadena de L_1 con una cadena de L_2. La concatenación de lenguajes constituye un *monoide*.

$$L_1 \cdot L_2 = \{ xy: x \in L_1 \wedge y \in L_2\}$$

También se define la **exponenciación** como simplificación notacional de la concatenación, con las siguientes propiedades:
- $L^n = L \cdot L^{n-1} = L^{n-1} \cdot L$
- $L^n \cdot L^m = L^{n+m}$
- $L^0 = \{\varepsilon\}$

Operaciones sobre lenguajes (3)

- **Iteración, cierre o clausura de un lenguaje**: El cierre de un lenguaje L (o *cerradura de Kleene*) se representa por L^* y se define como:

$$L^* = \{\varepsilon\} \cup L \cup L^2 \cup L^3 \cup \cdots = \bigcup_{i=0}^{\infty} L^i$$

- **Clausura positiva**: Se representa por L^+ y se define como:

$$L^+ = L \cup L^2 \cup L^3 \cup \cdots = \bigcup_{i=1}^{\infty} L^i$$

- Ejemplos: Si L={*a*, *b*, ...*z*, *A*, *B*, ...*Z*} y D={0, 1, ...9}, entonces:

  $L \cdot D$ es el lenguaje {*a*0, *a*1, ...*a*9, *b*0, ...*b*9, ...*Z*8, *Z*9}

  $L \cdot (L \cup D)^*$ es el lenguaje de las palabras que comienzan por una letra y pueden tener tanto letras como dígitos, es decir, el formato típico de los identificadores de los lenguajes de programación.

Gramáticas (1)

- El estudio de los lenguajes se puede hacer desde tres puntos de vista:
  1. el del **reconocimiento** del lenguaje: esta muy ligado a la teoría de autómatas y es "*...el estudio de algoritmos o estructuras de máquinas que permiten, dado un lenguaje L y una cadena x, determinar si* $x \in L$ *o* $x \notin L$*"* [Fernández, 1995]
  2. el de la **interpretación**: tiene que ver con la semántica de los lenguajes intenta formalizar el significado de las sentencias de un lenguaje.
  3. la **generación** de lenguajes: consiste en encontrar mecanismos que permitan enumerar las cadenas que pertenecen a un lenguajes. Estos mecanismos son las *gramáticas*.

Gramáticas (2)

- **Gramática formal**: Es una cuádrupla (Σ, N, P, S), donde:
  - Σ es un alfabeto (*conjunto finito* y *no vacío*) de **símbolos terminales** (o **tokens/tokenes**).
  - N es un alfabeto cuyos elementos se llaman **símbolos no terminales**
  - P es un alfabeto de **producciones** de la forma:

    $\{u \rightarrow v: u = xAy \in (\Sigma \cup N)^+ \wedge A \in N \wedge v \in (\Sigma \cup N)^*\}$
  - $S \in N$ es un símbolo especial llamado **axioma** o **símbolo inicial**.

Gramáticas (3)

- **Producción**: Una producción (o *regla de reescritura*) es un par ordenado de cadenas sobre un alfabeto ((x,y): $x, y \in \Sigma^*$). Se representa por $x \rightarrow y$. x es la parte izquierda o **antecedente** de la producción e y es la parte derecha o **consecuente**.
- **Derivación directa**: Sea $x \rightarrow y$ una producción y v, $w \in (\Sigma \cup N)^*$. Se dice que w *deriva directamente* de v, y se escribe $v \Rightarrow w$ si y sólo si existen z, $u \in (\Sigma \cup N)^*$ tales que $v = zxu$, $w = zyu$ y $x \rightarrow y$.
- **Derivación (en uno o más pasos)**: w deriva de v y se escribe $v \Rightarrow^+ w$ cuando existen $u_0, u_1, ..., u_n \in (\Sigma \cup N)^*$ tales que:

$$
\begin{aligned}
& v = u_0 \\
& u_0 \Rightarrow u_1 \\
& u_1 \Rightarrow u_2 \\
& \vdots \\
& u_{n-1} \Rightarrow u_n \\
& u_n = w
\end{aligned}
$$

A la secuencia $u_0, u_1, ..., u_n$ se la llama **cadena de derivación** de longitud n.

Gramáticas (4)

- **Derivación más a la izquierda**: cuando las producciones utilizadas en la derivación se aplican siempre a los símbolos más a la izquierda.
- **Derivación más a la derecha**: cuando las producciones se aplican a los símbolos más a la derecha.
- **Forma sentencial** de una gramática: Dada una gramática (Σ, N, P, S). Una cadena $\alpha \in (\Sigma \cup N)^*$ es una *forma sentencial* de esa gramática si existe una derivación que produce α a partir del axioma S, es decir si $S \Rightarrow^* \alpha$.
- **Frase** o **sentencia** de una gramática: Es una forma sentencial α que sólo contiene símbolos terminales ($\alpha \in \Sigma^*$).
- **Lenguaje generado por una gramática**: El *lenguaje generado por una gramática G* se representa por $L(G)$ y se define como el conjunto de todas las sentencias de la gramática G.

$$L(G) = \{x \in \Sigma^*: S \Rightarrow^+ x\}$$

Gramáticas (5)

- Dos gramáticas G_1 y G_2 se dice que son **gramáticas equivalentes**, y se representa por $G_1 \equiv G_2$, si generan el mismo lenguaje $L(G_1) = L(G_2)$.
- Una gramática G se dice que es *recursiva en un cierto símbolo no terminal U*, cuando existe una forma sentencial de U que contiene a U.

$$U \Rightarrow^+ xUy \quad \text{con } x, y \in (\Sigma \cup N)^*$$

- La **gramática** será **recursiva** si es recursiva para algún no terminal.
  - Si $x = \varepsilon$ se dice que es una **gramática recursiva a izquierdas**.
  - Si $y = \varepsilon$ se dice que es una **gramática recursiva a derechas**.

Árbol de derivación o sintáctico

- Son una forma de representar las derivaciones de algunas gramáticas.
- Se utilizan en los compiladores para representar el análisis sintáctico de los programas fuente y sirviendo de base para la generación de código.
- En los árboles de derivación:
  - el axioma se representa en la raíz del árbol;
  - los nodos hojas del árbol son símbolos terminales de la gramática; y
  - las derivaciones se representan creando tantos sucesores del símbolo no terminal de la izquierda de las producción como símbolos (terminales y no terminales) aparezcan en la parte derecha de las producciones.

Ambigüedad

La ambigüedad puede surgir a varios niveles: en sentencias, lenguajes, y gramáticas. A la hora de utilizar eficientemente los lenguajes y gramáticas, es conveniente que no exista ambigüedad, pues provoca que el análisis de las sentencias no sea determinista.

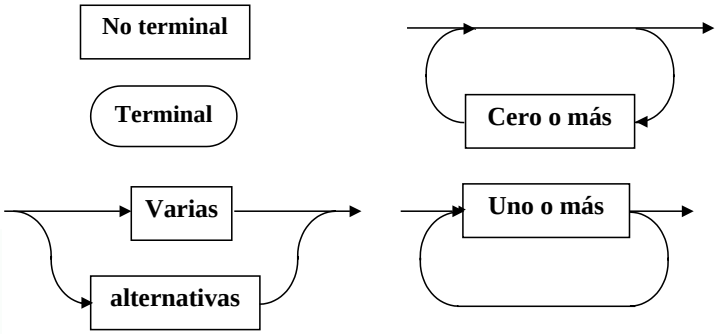
- **Sentencia**: una sentencia es ambigua si tiene más de árbol de derivación.
  Ej: En el caso de la gramática G=({**1**},{*S*, *B*},{*A*→ **1***B*, *A* →**11**, *B*→**1**}, *A*) la sentencia **11** puede ser obtenida por las dos derivaciones siguientes: *A*⇒**1***B*⇒**11** y *A*⇒**11**
- **Gramática**: una gramática es ambigua si tiene al menos una sentencia ambigua.
- **Lenguaje**: un lenguaje es ambiguo si existe una gramática ambigua que lo genera. Si todas las gramática que generan un lenguaje son ambiguas, el lenguaje es *inherentemente ambiguo* ($L=\{a^i b^j c^k: i=j \text{ o } j=k\}$).

Notación gramatical. Notación BNF.

- No es más que una forma de escribir las producciones. Las reglas son:
  ① Los símbolos terminales se suelen poner en negrita, y los no terminales entre ángulos < >
  ② El nexo para la producción es **::=**
  ③ La agrupación de reglas mediante |
  ④ Una o más veces mediante corchetes **[ ]**
  ⑤ Cero o más veces mediante llaves **{ }**
  ⑥ Paréntesis para delimitar casos de duda en el uso de los operadores meta-lingüísticos

Notación gramatical. Diagramas sintácticos

- Los diagramas sintácticos (diagramas de trenes, o diagramas de Conway), son construcciones gráficas que permiten presenta la misma información sintáctica que la notación BNF pero de una manera más fácil de interpretar para las personas.



Clasificación de gramáticas (1)

- **Gramáticas de tipo 0** o **gramáticas de Chomsky**, con reglas de producción de la forma:

  $u \rightarrow v$ con $u = xAy \in (\Sigma \cup N)^+ \wedge A \in N \wedge x,y,v \in (\Sigma \cup N)^*$

- El conjunto de lenguajes de tipo 0 coincide con el de todos los lenguajes gramaticales posibles.
- Puede demostrarse que todo lenguaje generado por una gramática de Chomsky puede generarse también por unas gramáticas más restrictivas llamadas **gramáticas con estructura de frase**, cuyas reglas de producción son de la forma:

  $xAy \rightarrow xvy$ con $x,y,v \in (\Sigma \cup N)^* \wedge A \in N$

Clasificación de gramáticas (2)

- **Gramáticas de tipo 1** o **sensibles al contexto**, con reglas de producción de la forma:

  $xAy \rightarrow xvy$ con $x,y \in (\Sigma \cup N)^* \wedge A \in N \wedge v \in (\Sigma \cup N)^+$

- En los lenguajes generados por estas gramáticas el significado de las "palabras" depende de su posición en la frase.
- A la *x* e *y* es a lo que se le llama **contexto** (es decir, *A* sólo puede transformarse en *v* si va precedido de *x* y seguido de *y*).
- No tiene *reglas compresoras*, aunque se tolera la regla $S \rightarrow \varepsilon$.
- Son las gramáticas de mayor categoría que se suelen utilizar (la mayor parte de los lenguajes de ordenador pertenece a este grupo, aunque gran parte de las reglas de las gramáticas que los generan pueden reducirse a las de tipo 2).
- Se caracteriza por que la longitud de las formas sentenciales partiendo de *S* es siempre **no** decreciente.

Clasificación de gramáticas (3)

- **Gramáticas de tipo 2** o **independientes del contexto**, con reglas de la forma:

  $A \rightarrow v$ con $A \in N \wedge v \in (\Sigma \cup N)^*$

- Se vuelven a introducir leyes compresoras, pero es fácil demostrar que se puede obtener una gramática equivalente que no las tenga, obteniéndose una definición algo más restrictiva:

  $A \rightarrow v$ con $A \in N \wedge v \in (\Sigma \cup N)^+$
  además es posible que se tenga la regla $S \rightarrow \varepsilon$

- En los lenguajes generados por las gramáticas de este tipo el significado de las "palabras" es independiente de su posición.
- Una última característica de este tipo de gramáticas es que las derivaciones obtenidas al utilizarlas se pueden representar utilizando árboles.

Clasificación de gramáticas (4)

- **Gramáticas de tipo 3** o **gramáticas regulares**, con reglas de la forma:

  $A \rightarrow aB \wedge A \rightarrow b$ o de la forma $A \rightarrow Ba \wedge A \rightarrow b$
  con $A,B \in N \wedge a,b \in \Sigma$

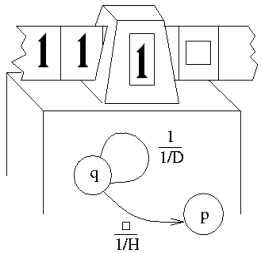
- A las gramáticas regulares del primer tipo se las llama **gramática regulares a derechas**, a las del segundo tipo **gramáticas regulares a izquierdas**, en realidad son totalmente equivalentes. Si ε pertenece al lenguaje, se tolera la regla $S \rightarrow \varepsilon$.
- Existe una generalización de este tipo de gramáticas llamadas **gramáticas lineales** con reglas de la forma:

  $A \rightarrow aB \wedge A \rightarrow b$ o de la forma $A \rightarrow Ba \wedge A \rightarrow b$
  con $A,B \in N \wedge a,b \in \Sigma^*$

- son totalmente equivalentes a las gramáticas regulares normales, pero en muchos casos su notación es más adecuada.

Máquinas de Turing (1)

- Una máquina de Turing es un dispositivo capaz de adoptar un estado determinado, conectado a una cabeza de E/S, que puede leer y escribir un símbolo en una cinta de tamaño infinito.
- En función del símbolo leído y del estado:
  1. Pasa a un nuevo estado,
  2. imprime un símbolo en la cinta, y
  3. mueve la cabeza a la izquierda (I) a la derecha (D) o se para (P).



Máquinas de Turing (2)

- **Máquina de Turing**: Es una 7-tupla ($\Gamma,\Sigma,b,Q,q_0,F,f$), donde:
  - Γ es el alfabeto de elementos llamados: **símbolos** o **caracteres de pila**.
  - Σ es un alfabeto de **símbolos de entrada** ($\Sigma\subseteq\Gamma-\{b\}$).
  - b símbolo especial de Γ ($b\in\Gamma$ y $b\notin\Sigma$).
  - Q es un alfabeto de **estados**.
  - q_0 es un elemento de Q ($q_0\in Q$).
  - F es un subconjunto de Q de **estados finales** o de **aceptación** ($F\subseteq Q$).
  - f es una función de la forma:

$$Q,\times\Gamma\rightarrow Q,\times\Gamma\times\{I, D, P\}$$

Máquinas de Turing (3)

- Para determinar la situación en que se encuentra una máquina de Turing es preciso saber:
  1. El estado.
  2. Toda la información de la cinta.
  3. La posición de la cabeza.
- Se puede tener una **descripción instantánea** de la máquina de Turing usando una representación monodimensional en la que la posición de la cabeza y el estado se representa poniendo el nombre del estado a la izquierda del símbolo sobre el que se encuentra la cabeza.

Máquinas de Turing (4)

- Considérese una máquina de Turing

$$T=\{\Sigma\cup\{\blacktriangleleft,\blacktriangleright,\square\},\Sigma\cup\{\blacktriangleleft,\blacktriangleright\},\square,Q,q_0,F,f\}$$

tal que:

1. La cinta de entrada contiene la palabra: $\blacktriangleleft x\blacktriangleright$ con $x\in\Sigma^*$
2. La cabeza de lectura/escritura se encuentra inicialmente sobre el símbolo: $\blacktriangleleft$

- Se dice que ***x* es aceptada por *T*** si la máquina se detiene en un estado $q\in F$ y la cinta queda totalmente borrada.
- Por tanto se define el **lenguaje *L* reconocido por una máquina de Turing *T*** como:

$$L=\{x\in\Sigma^* \text{ tales que } x \text{ es aceptado por } T\}$$

Máquinas de Turing (5)

- El proceso de reconocimiento es inverso al de generación, por tanto va a ser posible construir la gramática de un lenguaje a partir de la máquina de Turing que lo reconoce.
- Dada una máquina de Turing:
  $T=\{\Sigma\cup\{\triangleleft,\triangleright,\square\},\Sigma\cup\{\triangleleft,\triangleright\},\square,Q,q_0,F,f\}$
- La gramática que genera el lenguaje que T reconoce será de la forma:
  $T=\{\Sigma, Q\cup\{S,(,),\triangleleft,\triangleright,\square\},S,P\}$
- Donde P contiene las reglas de producción obtenidas al aplicar determinadas reglas.

Máquinas de Turing (6)

- Una producción paso del axioma a la descripción instantánea final de la máquina de Turing.
  - $S\rightarrow(q\square)\ \ \forall q\in F$
- Por cada transición de f de la forma $f(p,a)=(q,b,D)$ las producciones:
  - $b\mathbf{q}\rightarrow\mathbf{p}a$
  - $(\rightarrow(\square$
  - $b\mathbf{q}\square)\rightarrow\mathbf{p}a)$
- Por cada transición de f de la forma $f(p,a)=(q,b,I)$ las producciones:
  - $(\mathbf{q}\square b\rightarrow(\mathbf{p}a$
  - $\mathbf{q}cb\rightarrow c\mathbf{p}a \qquad \forall c\in\Sigma\cup\{\triangleleft,\triangleright\}$
  - $a)\rightarrow a\square) \qquad \forall a\in\Sigma\cup\{\triangleleft,\triangleright\}$
- Por cada transición de f de la forma $f(p,a)=(q,b,H)$ las producciones:
  - $\mathbf{q}b\rightarrow\mathbf{p}a$
- Las reglas anteriores nos permite llegar a $(q_0\triangleleft x\triangleright)$ ya sólo falta eliminar los no terminales dejando solo x.
  - $(\mathbf{q_0}\triangleleft a\rightarrow a \qquad \forall a\in\Sigma$
  - $a\triangleright)\rightarrow a \qquad \forall a\in\Sigma$

Variantes de la máquina de Turing

- Máquina de Turing “escribir o mover”
- Máquina de Turing “mover o permanecer”
- Máquina de Turing con cinta limitada en un sentido
- Máquina de Turing binaria
- Máquina de Turing multicabeza
- Máquina de Turing multipista
- Máquina de Turing multicinta
- Máquina de Turing multidimensional
- Máquina de Turing no determinista

Máquina de Turing Universal

- Se trata de una máquina de Turing que toma como entrada la codificación de las transiciones de otra y la entrada sobre la que debe actuar la máquina, y simula su funcionamiento.
- Primero hay que dar un procedimiento que permita codificar las transiciones de una máquina de Turing, y por tanto la propia máquina.

Otros autómatas

- Máquina linealmente acotada
  - se puede ver como una máquina de Turing que sólo puede utilizar una porción acotada de la cinta.
- Autómata de pila
  - se puede ver como una MT que sólo puede leer la cinta en un sentido, aunque puede utilizar una pila para almacenar lo que ha leído.
- Autómata finito
  - se puede ver como una MT que sólo puede leer la cinta en un sentido y no tiene ningún tipo de dispositivo de almacenamiento (salvo el propio estado interno en el que se encuentra).
- Redes de Petri.
- Autómatas estocásticos o probabilistas.
- Autómatas borrosos.
- Autómatas celulares.

Relación entre autómatas y lenguajes

